NTNU

Det skapende universitet

TDT4110 Informasjonsteknologi grunnkurs:

Tema: Filer og unntak (”exceptions”)

- 3rd edition: Kapittel 6

Professor Alf Inge Wang

Læringsmål og pensum

Mål

- Lære bruk av inn- og utoperasjoner i Python
- Lære å kunne bruke lesing og skriving til fil
- Lære å kunne unntak (Exceptions)

Pensum

- Starting out with Python,Chapter 6 ”Files and Exceptions”

Inn- og utoperasjoner

- I programmering brukes begrepet inn/utoperasjoner, I/O eller IO når programmet leser eller skriver innholdet av variabler (data) til omverdenen.
  - Omverdenen kan være skjermen (ut), tastaturet (inn), skriver (ut), eller til filer (inn og ut)
- Bruk av tastatur og skjerm er ok for små datamengder, men upraktisk for store datamengder:
  - Eks. værdata, modeller av skip eller konstruksjon, osv. som kan bestå av milliarder av bytes (gigabytes) eller mer.
  - Eks. data fra seismiske undersøkelser kan være i størrelsesorden pentabytes (en million milliarder bytes).

Inn- og utoperasjoner (2)

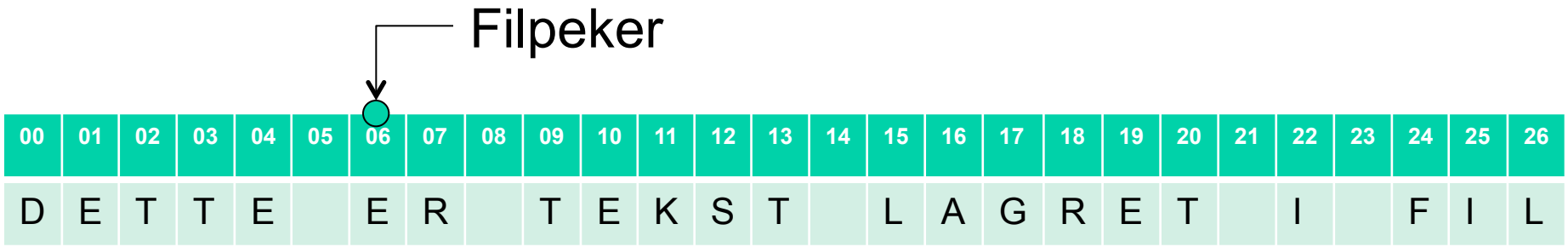
- Ved kun bruk av i/o-operasjoner mot tastatur og skjerm, vil all data forsvinne etter vi har avsluttet Python.
- Å lagre filer som inneholder data til sekundærminne (for eksempel harddisk), gjør det mulig å huske data mellom hver gang man kjører/avslutter programmer.
- Vi har allerede lagret programmer i Python-filer som vi kan gjenbruke gang etter gang.
- Nå skal vi lære å gjøre det samme med dataene som brukes i programmer (innholdet i variabler).

Hva er en fil i Python?

- I et Python-program, blir ei fil representert som en verdi av typen *file.*
- Verdien til en fil representerer ikke innholdet i fila, men en referanse eller portal til dataen.
- En fil som er lagret på en harddisk kan inneholde mer data enn du kan ha i minnet på en gang.
  - Det er derfor viktig at du har referanse til fila og kan navigere deg igjennom en fil for å skrive eller hente data.
  - Man henter ofte in bare deler av ei fil og lagrer dette i variabler.

Hva er en fil i Python? (2)

- I en fil lagres data etter hverandre (sekvensielt).
- Etter hvert som man skriver data til en fil, vil en filpeker holde orden på hvor langt man har kommet i fila.
- Filpekeren kan også flyttes ved kommandoer.



NTNU
Det skapende universitet

Prosessen for filoperasjoner i Python

Filoperasjoner i Python gjøres i tre steg:

1. Fila åpnes: Etablerer en link mellom filvariabelen og informasjonen lagret på disken.

   Filreferansen som peker til den fysiske fila på disk blir lagret i en variabel.

   Alle operasjoner mot fila må bruke denne variabelen som filreferanse.

2. Verdier leses fra og skrives til fila ved hjelp av filreferansen:

   Lesing: Data lagret i fil leses inn og lagres i variabler.

   Skriving: Data lagret i variabler skrives som data lagret i en fil.

3. Fila stenges.

   Etter at fila er stengt, kan man ikke lese eller skrive til fila.

Filhåndtering i Python

Filkommandoer	Forklaring
f = open(‘filnavn’)	Åpner ei fil, returnerer filreferanse
f = open(‘filnavn’,’tilgang’)	Åpner ei fil, med spesifisert tilgang. F.eks. ‘w’ åper ei fil for skriving (se neste side)
f.read()	Returnerer hele innholdet av fila
f.read(n)	Returnerer n karakterer av innholdet
f.readline()	Returnerer neste linje (før \n)
f.readlines()	Returnerer hele innholdet av fila som ei liste
f.write(s)	Skriver ***strengen*** s til fil
f.writelines(liste)	Skriver innholdet av liste av ***strenger*** til fil
f.seek(offset,fra_hvor)	Forflytter filpekeren (index) i fila
f.tell()	Returnerer posisjon til filpekeren i fila
f.close()	Stenger fila

f representerer variabelen som tar vare på filpekeren

Åpning av filer

For å lagre data til fil eller hente data fra fil, må man først åpne fila ved hjelp av `open`:

```
variabel = open('filnavn' , 'tilgangstype')
```

Forklaring:

variabel: Får en referanse som peker til fila med angitt filnavn

filnavn: Angir et stinavn og filnavn til fila som skal åpnes

tilgangstype: Angir en kode for typen filoverførings som skal gjøres

Eks:

```
f = open('datafil.txt','r') # Fil for lesing
f = open('datafil.txt','w') # Fil for skriving
```

Tilgangstyper for fopen

Vi har følgende tilgangstyper for `open`:

Streng	Filoperasjon
'r'	Åpne en fil for *lesing*
'w'	Åpne en fil for *skriving* og fjern eventuelt gammelt innhold. Lag en ny fil hvis den ikke fins.
'a'	Åpne en fil for å *legge til* nye data på slutten (logging). Lag en ny fil hvis den ikke fins.
'r+'	Åpne en fil som fins fra før for *lesing og skriving*.
'w+'	Åpne en fil for *lesing og skriving*, og fjern eventuelt gammelt innhold. Lag en ny fil hvis den ikke fins.
'a+'	Åpne en fil for å *lese og legge til* nye data på slutten (logging). Lag en ny fil hvis den ikke fins.

Stenging av filer

- Når et program åpner en fil for lesing, vil operativsystemet vite at et program leser fila.
- Når et program åpner en fil for skriving (endring), låser operativsystemet fila slik at ingen andre får lov til å endre på fila samtidig.
- Etter programmet er ferdig med å lese fra og skrive til fila, må fila lukkes for å si ifra at nå er den fritt vilt:
- `filvariabel.close() # Stenger fila`

Skrive strenger til fil

- For å skrive data til fil i Python brukes følgende:

```
f.write(s)              # Skriver strengen s til fil med ref f
f.writelines(liste)     # Skriver en liste av strenger til fil
```

- Vi ser på et program der brukeren kan skrive inn navnet på fila hvor tekst brukeren skriver inn blir lagret.
- Brukeren avslutter skriving med linjeskift (uten tekst)

- Vi prøver…

skriv_til_fil.py

Skrive liste av strenger til fil

- En liste av strenger kan skrives til fil ved hjelp av:

  ```
  filvariabel.writelines(liste)
  ```

- Vi lager en variant av skriv_til_fil.py der vi bruker liste. Må da:
  - Opprette en tom liste
  - `append` tekst-strengen brukeren skriver inn til lista
  - Skrive lista til fil

- Vi prøver…

skriv_linjer_til_fil.py

Skrive annen data til fil

- Merk at det er kun strenger som skrives til fil.
- For å skrive annen data en strenger, så må man konvertere dette til streng:
  - Kan bruke str(variabel)
- Vi ser på et eksempel for å lagre masse tall en bruker skriver inn til en fil med filnavn spesifisert av brukeren.

- Vi ser på et eksempel…

skriv_tall_til_fil.py

Lese strenger fra fil

- For å lese strenger fra fil, benyttes:

```
streng = filvariabel.read() # returnerer hele innholdet
streng = filvariabel.readline() # returnerer ei linje
```

- `read()` kan benyttes hvis fila ikke inneholder for store mengder data, men bør unngås for store filer.
- `readline()` gjør det mulig å gå igjennom fila linje for linje, men krever at fila er oppdelt med linjeskift (\n).
  - Kan bruke while-løkke for å sjekke om vi har kommet til enden.
- Vi ser på to eksempler:

les_fil.py les_linjer_fil.py

Lese som liste av strenger fra fil

- Vi kan også få returnert innholdet av ei tekstfil som ei liste av strenger:

```
liste = filvariabel.readlines() # returnerer liste
```

- `readlines()` kan være veldig praktisk hvis man skal utføre listeoperasjoner på innholdet i fila, for eksempel highscore liste for dataspill.
- Vi ser på et eksempel:

les_liste_fil.py

Lese tall fra fil

- Når vi skal lese inn tall fra en tekstfil i Python, må vi oversette dataen fra streng til et tall:

```
variabel = int(streng) # heltall
variabel = float(streng) # flyttall
variabel = eval(streng) # uttrykk/tall
```

Å bruke Pythons for-løkke til å lese linjer

- Python tillater å skrive en for-løkke som automatisk leser linjer fra fil og slutter for-løkka når den når slutten av fila:
  - Format:
```
for line in file_object:
    kode..
```
  - Løkka går igjennom (itererer) fila linje for linje

Oppgave: les tall fra fil

- Skriv Python-koden for å gjøre følgende:
  - Spør bruker om filnavn
  - Åpne fila for lesing
  - Les inn fra fil, linje for linje ved hjelp av for-løkke
  - Konverter fra streng til tall
  - Skriv ut tallet lest inn fra fil opphøyd i tredje

les_tall_fil.py

Lese fra ei fil, tegn for tegn

- Det er mulig å spesifisere at vi skal lese ett gitt antall tegn i gangen fra en fil. Dette gjøres ved:

```
tegn = filvariabel.read(n) # n er antall tegn
```

- Dette gjøre det mulig å for eksempel søke etter et spesielt tegn inne i fila.
- Vi ser på et eksempel der brukeren kan skrive inn et tegn som det skal søkes etter i ei fil med filnavn som brukeren også skriver inn:

les_tegn_for_tegn.py

Nyttige funksjoner for filbehandling i bilioteket os

- Biblioteket *os* gir en del nyttige funksjoner for filbehandling.
- Først må man skrive: `import os`
- Få ei liste av alle filer i filkatalogen:

```
liste = os.listdir()
```

- Endre nåværende katalog/directory:

```
os.chdir(path)
```

- Finne ut hva som er nåværende file path/fil sti:

```
variable = os.getcwd()
```

Unntak (“Exceptions”)

Kapittel 6.4

Exception / Unntak

- En exception er en feil som oppstår under kjøring som får programmet til å stoppe opp.
- Typiske feil som gir exception er:
  - Prøver å gjøre om tekststrenger til tall med strenger uten tall
  - Divisjon på 0
  - Prøver å åpne filer som ikke eksisterer
- En måte å unngå dette er å sjekke bruker-input.
- I Python kan du også bruke `try/exception` uttrykk for å unngå at programmet stopper opp under slike feil.

Exceptions / Unntak

- Exception: feil som skjer når et program kjører
  - Som regel fører det fører det til at programmet stopper (kræsjer)
- Exception handling: Håndtere ”exceptions” ved å gi brukeren fornuftig tilbakemelding uten at programmet stopper helt opp.
- Benytter:

```
try:        # Prøv om koden lar seg kjøre
except      # Fanger opp hvis koden i try feiler
except Exception as variable: # fanger feilmelding
else:       # Kjøres hvis det ikke blir exception
finally:    # Kjøres uansett til slutt
```

Exception: try – except uttrykk

- "Usikker" kode skrives inne i et `try:` uttrykk
  - Tester ut om denne koden kjører uten problemer
- I tillegg må vi legge til kode som fanger opp eventuelle feil: `except ExceptionName:`

```
try:            # En feil i try-blokka, trigger except
    uttrykk
    uttrykk
    …
except ExceptionName: # Hopper hit hvis feil i try
    uttrykk
    uttrykk
```

NTNU
Det skapende universitet

Exception – ExceptionName

- Ulike typer Exceptions har ulike navn.
- Vi kan fange opp disse ved å lage en exception i kode.
- Typiske ExceptionName er:
  - ValueError: Typisk feil i datatype (streng når det skal være tall)
  - ZeroDivisionError: Prøver å dividere med 0
  - IOError: Feil med filbehandling
  - Exception: Alle mulige feil (generell)
- Ser på et eksempel på bruk av try – except:

try_except.py

Exception – vis innebygd feilmelding

- Det er mulig å fange opp feilmeldingen som Python gir ved en Exception ved bruk av følgende kode:

```
try:
    uttrykk…
except Exception as variabel:
    print(variabel)
```

- Uttrykket `as variabel`, fanger opp feilen og lagrer feilmeldingen i en variabel som opprettes.
- Vi ser på et eksempel:

exception_vis_feilmelding.py

NTNU
Det skapende universitet

Exception – else og finally

- Et `try – except` uttrykk kan også bestå av `else` og `finally`:
- `else` blir utført hvis ingen exceptions ble trigget.
- `finally` blir utført til slutt uansett om exceptions ble trigget eller ikke

```
try:
        uttrykk…
except ExceptionName:
    uttrykk…
else:
    uttrykk…
finally:
    uttrykk…
```

exception_finally.py

Oppsummering

- Filhåndteringsprosess:
  - Åpner en fil med en gitt aksess
  - Leser fra fil/skriver til, evt. forflytter filpeker
  - Lukker fil
- Vi kan jobbe med flere filer samtidig:
  - Filvariabelen med referanse til fila bestemmer hvilken fil vi jobber med.